CLAIMS

1. A computer system providing an object-based virtual machine environment for running successive applications, said computer system including storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said system including:

a card table comprising multiple cards, each corresponding to a region of said storage, each card in the card table being set to null when the first heap is reset between successive applications;

means for marking a card whenever an object in its corresponding storage region is updated; and

means for detecting possible references from the second heap to the first heap at reset by scanning the cards in the card table corresponding to the second heap, and detecting any cards which have been marked.

2. The computer system of claim 1, further comprising:

means for locating, for each marked card, any objects in the corresponding region of storage; and

means for identifying any references to the first heap in the located objects.

3. The computer system of claim 2, further comprising:

means responsive to the identification of references to the first heap for performing the mark phase of a garbage collection to determine live objects in at least the second heap;

means for detecting whether any objects in the second heap having references to the first heap have been marked as live; and

means responsive to a detection of any such objects for returning an error condition to prevent reset for another application.

4. The computer system of claim 3, further comprising means for invalidating the card table if a compact operation has been performed on the second heap since the last reset, wherein said means for performing the mark phase is also responsive to invalidation of the card table.

5. The computer system of claim 1, wherein an object is only considered as within the region of storage corresponding to a card if a predetermined part of the object is in that region.

6. The computer system of claim 1, wherein the region of memory corresponding to a card comprises between 256 and 2048 bytes.

7. The computer system of claim 1, further comprising:

means for detecting references or possible references to the first heap from a set of predetermined locations; and

means responsive to the detection of any such references or possible references for returning an error condition to prevent reset for another application.

8. The computer system of claim 7, wherein said set of predetermined locations includes the stacks and registers.

9. The computer system of claim 1, further comprising:

means for detecting any objects on the first heap which are reachable from virtual machine system class objects; and

means for promoting any such detected objects to the second heap.

10. A computer system providing an object-based virtual machine environment for running successive applications, said computer system including storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said system including:

means for identifying any objects on the first heap which have a finalization method; and

means for running the finalization methods of any identified objects on the main thread prior to reset of the first heap.

11. The computer system of claim 10, further comprising means responsive to running said finalization methods for checking that they have not performed any operations which would prevent reset of the first heap.

12. A method of operating a computer system providing an object-based virtual machine environment for running successive applications, said computer system including

storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said method including the steps of:

providing a card table comprising multiple cards, each corresponding to a region of said storage, each card in the card table being set to null when the first heap is reset between successive applications;

marking a card whenever an object in its corresponding storage region is updated; and

detecting possible references from the second heap to the first heap at reset by scanning the cards in the card table corresponding to the second heap, and detecting any cards which have been marked.

13. The method of claim 12, further comprising:

locating, for each marked card, any objects in the corresponding region of storage; and

identifying any references to the first heap in the located objects.

14. The method of claim 13, further comprising:

responsive to the identification of references to the first heap, performing the mark phase of a garbage collection to determine live objects in at least the second heap;

detecting whether any objects in the second heap having references to the first heap have been marked as live; and

responsive to a detection of any such objects, returning an error condition to prevent reset for another application.

15. The method of claim 14, further comprising the step of invalidating the card table if a compact operation has been performed on the second heap since the last reset, wherein said step of performing the mark phase is also performed in response to invalidation of the card table.

16. The method of claim 12, wherein an object is only considered as within the region of storage corresponding to a card if a predetermined part of the object is in that region.

17. The method of claim 12, wherein the region of memory corresponding to a card comprises between 256 and 2048 bytes.

18. The method of claim 12, further comprising:

detecting references or possible references to the first heap from a set of predetermined locations; and

responsive to the detection of any such references or possible references, returning an error condition to prevent reset for another application.

19. The method of claim 18, wherein said set of predetermined locations includes the stacks and registers.

20. The method of claim 12, further comprising:

detecting any objects on the first heap which are reachable from virtual machine system class objects; and

promoting any such detected objects to the second heap.

21. A method of operating a computer system providing an object-based virtual machine environment for running successive applications, said computer system including storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said method including the steps of:

identifying any objects on the first heap which have a finalization method; and

running the finalization methods of any identified objects on the main thread prior to reset of the first heap.

22. The method of claim 21, further comprising the step, responsive to running said finalization methods, of checking that they have not performed any operations which would prevent reset of the first heap.

23. A computer program product for operating a computer system providing an object-based virtual machine environment for running successive applications, said computer system including storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said computer program product comprising machine-readable program instructions recorded on a storage medium, said

instructions when loaded into the computer system causing it to perform the steps of:

providing a card table comprising multiple cards, each corresponding to a region of said storage, each card in the card table being set to null when the first heap is reset between successive applications;

marking a card whenever an object in its corresponding storage region is updated; and

detecting possible references from the second heap to the first heap at reset by scanning the cards in the card table corresponding to the second heap, and detecting any cards which have been marked.

24. The computer program product of claim 23, wherein said program instructions further cause the computer to perform the steps of:

locating, for each marked card, any objects in the corresponding region of storage; and

identifying any references to the first heap in the located objects.

25. The computer program product of claim 24, wherein said program instructions further cause the computer to perform the steps of:

responsive to the identification of references to the first heap, performing the mark phase of a garbage collection to determine live objects in at least the second heap;

detecting whether any objects in the second heap having references to the first heap have been marked as live; and

responsive to a detection of any such objects, returning an error condition to prevent reset for another application.

26. The computer program product of claim 25, wherein said program instructions further cause the computer to perform the step of invalidating the card table if a compact operation has been performed on the second heap since the last reset, wherein said step of performing the mark phase is also performed in response to invalidation of the card table.

27. The computer program product of claim 23, wherein an object is only considered as within the region of storage corresponding to a card if a predetermined part of the object is in that region.

28. The computer program product of claim 23, wherein the region of memory corresponding to a card comprises between 256 and 2048 bytes.

29. The computer program product of claim 23, wherein said program instructions further cause the computer to perform the steps of:

detecting references or possible references to the first heap from a set of predetermined locations; and

responsive to the detection of any such references or possible references, returning an error condition to prevent reset for another application.

30. The computer program product of claim 29, wherein said set of predetermined locations includes the stacks and registers.

31. The computer program product of claim 23, wherein said program instructions further cause the computer to perform the steps of:

detecting any objects on the first heap which are reachable from virtual machine system class objects; and

promoting any such detected objects to the second heap.

32. A computer program product for operating a computer system providing an object-based virtual machine environment for running successive applications, said computer system including storage, at least a portion of which is logically divided into two or more heaps in which objects can be stored, wherein a first heap is reset between successive applications, and a second heap persists from one application to the next, said computer program product comprising machine-readable program instructions recorded on a storage medium, said instructions when loaded into the computer system causing it to perform the steps of:

identifying any objects on the first heap which have a finalization method; and

running the finalization methods of any identified objects on the main thread prior to reset of the first heap.

33. The computer program product of claim 21, wherein said program instructions further cause the computer to perform the step responsive to running said finalization methods, checking that they have not performed any operations which would prevent reset of the first heap.